第5章

回　　归

回归是机器学习的核心算法之一。回归基于统计原理，对大量统计数据进行数学处理，并确定变量（或属性）之间的相关关系，建立一个相关性的回归方程（函数表达式），用于预测今后的因变量。

5.1　回归概述

5.1.1　回归的概念

分类算法用于离散型分布预测，前面讲过的决策树、朴素贝叶斯、支持向量机都是分类算法；回归算法用于连续型分布预测，针对的是数值型的样本。“回归”用于表明一个变量的变化，会导致另一个变量的变化，即有前因后果的变量之间的相关关系。回归分析研究某一随机变量（因变量）与其他一个或几个普通变量（自变量）之间的数量变动的关系。回归的目就是建立一个回归方程来预测目标值。回归的求解就是求这个回归方程的参数。

5.1.2　回归处理流程

回归分析的基本思路是：从一组样本数据出发，确定变量之间的数学关系式，对这些关系式的可信程度进行各种统计检验，并从影响某一特定变量的诸多变量中找出哪些变量的影响显著，哪些不显著。然后利用所求的关系式，根据一个或几个变量的取值预测另一个特定变量的取值。

5.1.3　回归的分类

根据自变量数目的多少，回归模型可以分为一元回归模型和多元回归模型；根据模型中自变量与因变量之间是否线性，回归模型可以分为线性回归模型和非线性回归模型；根据模型中是否带有虚拟变量，回归模型可以分为普通回归模型和带虚拟变量的回归模型。

5.2 一元线性回归

5.2.1 一元线性回归介绍


一元线性回归

一元线性回归(linear regression)只研究一个自变量与一个因变量之间的线性关系,一元线性回归模型可表示为 $y=f(x;w,b)=wx+b$,其图形为一条直线。用数据寻找一条直线的过程也叫作拟合一条直线。为了选择在某种方式下最好的 w 和 b 值,需要定义模型最好的意义是什么。所谓最好的模型,由 w 和 b 的一些值组成,这些值可以产生一条能尽可能与所有数据点接近的直线。衡量一个特定的线性模型 $y=wx+b$ 与数据点接近程度的普遍方法是真实值与模型预测值之间差值的平方:

$$(y_1-(wx_1+b))^2$$

这个数值越小,模型在 x_1 处越接近 y_1。这个表达式称为平方损失函数,因为它描述了使用 $f(x_1;w,b)$模拟 y_1 所损失的精度,在本章中,用 $L_n()$表示损失函数,在这种情况下:

$$L_n(y_n,f(x_n;w,b))=(y_n,f(x_n;w,b))^2$$

这是第 n 个点处的损失。损失总是正的,并且损失越小,模型描述这个数据就越好。对于所有 n 个样本示例,想有一个低的损失,可考虑在整个样本示例集上的平均损失(均方误差),即

$$L=\frac{1}{n}\sum_{i=1}^{n}L_i(y_i,f(x_i;w,b))$$

这是 n 个样本示例的平均损失值,该值越小越好。因此,可通过调整 w 和 b 值产生一个模型,此模型得到的平均损失值最小。寻找 w 和 b 的最好值,用数学表达式可以表示为

$$(w^*,b^*)=\underset{(w,b)}{\operatorname{argmin}}=\frac{1}{n}\sum_{i=1}^{n}L_i(y_i,f(x_i;w,b))$$

argmin 是数学上“找到最小化参数”的缩写,w^*、b^* 表示 w 和 b 的解。

将均方误差作为模型质量评估的损失函数有非常好的几何意义,它对应了常用的欧几里得距离,简称“欧氏距离”。基于均方误差最小化进行模型参数求解的方法称为“最小二乘法”。在线性回归中,最小二乘法就是试图找到一条直线 $y=wx+b$,使所有样本到直线的欧氏距离之和最小。

求解 w 和 b 使 $E(w,b)=\sum_{i=1}^{n}(y_i-(wx_i+b))^2$ 最小化的过程,称为线性回归模型的最小二乘“参数估计”,为此,分别求 $E(w,b)$对 w 和 b 的偏导并令它们等于 0,求这两个方程就可以求出符合要求的待估参数 w 和 b:

$$w=\frac{n\sum_{i=1}^{n}x_iy_i-\sum_{i=1}^{n}x_i\sum_{i=1}^{n}y_i}{n\sum_{i=1}^{n}x_i^2-\left(\sum_{i=1}^{n}x_i\right)^2},\quad b=\frac{1}{n}\left(\sum_{i=1}^{n}y_i-w\sum_{i=1}^{n}x_i\right)$$

其中，n 为样本的数量；y_i 为样本的真实值。

【例 5-1】　设 10 个厂家的投入和产出见表 5-1。根据这些数据可以认为投入和产出之间存在线性相关性吗？

表 5-1　10 个厂家的投入和产出

厂家	1	2	3	4	5	6	7	8	9	10
投入	20	40	20	30	10	10	20	20	20	30
产出	30	60	40	60	30	40	40	50	30	70

下面给出使用 sklearn.linear_model 模块下的 LinearRegression 线性回归模型求解例 5-1 投入、产出所对应的回归方程的参数。

LinearRegression 线性回归模型的语法格式如下。

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

其功能是创建一个回归模型，返回值为 coef_和 intercept_，coef_存储 $b_1 \sim b_m$ 的值，与回归模型将来训练的数据集 X 中每条样本数据的维数一致，intercept_存储 b_0 的值，即线性回归方程的常数项。

参数说明如下。

copy_X：布尔型，默认为 True，用来指定是否对训练数据集 X 进行复制（即经过中心化、标准化后，是否把新数据覆盖到原数据上），如果选择 False，则直接对原数据进行覆盖。

fit_intercept：用来指定是否对训练数据进行中心化，如果该变量为 False，则表明输入的数据已经进行了中心化，在下面的过程里不进行中心化处理；否则，对输入的训练数据进行中心化处理。

n_jobs：整型，默认为 1，用来指定计算时设置的任务个数。如果选择－1，则代表使用所有的 CPU。

normalize：布尔型，默认为 False，用来指定是否对数据进行标准化处理。

LinearRegression 线性回归模型对象的主要方法有以下几个。

fit(X, y[, sample_weight])：对训练数据集 X、y 进行训练，sample_weight 为[n_samples]形式的向量，可以指定对于某些样本数据 sample 的权值，如果觉得某些样本数据 sample 比较重要，可以将这些数据的权值设置得大一些。

get_params([deep])：获取回归模型的参数。

predict(X)：使用训练得到的回归模型对 X 进行预测。

score(X, y[,sample_weight])：返回对以 X 为样本数据 samples，以 y 为实际结果 target 的预测效果评分，最好的得分为 1.0，一般的得分都比 1.0 低，得分越低，代表模型的预测结果越差。

set_params(* * params)：设置 LinearRegression 模型的参数值。

求解例 5-1 的线性回归模型的参数的代码如下。

```
import matplotlib.pyplot as plt
import matplotlib
import numpy as np
matplotlib.rcParams['font.family'] = 'FangSong'  #指定字体的中文格式
#定义一个画图函数
def runplt():
    plt.figure()
    plt.title('10 个厂家的投入和产出',fontsize=15)
    plt.xlabel('投入',fontsize=15)
    plt.ylabel('产出',fontsize=15)
    plt.axis([0,50,0,80])
    plt.grid(True)
    return plt
#投入、产出训练数据
X =[[20],[40],[20],[30],[10],[10],[20],[20],[20],[30]]
y =[[30],[60],[40],[60],[30],[40],[40],[50],[30],[70]]
from sklearn.linear_model import LinearRegression
model =LinearRegression()                          #建立线性回归模型
model.fit(X,y)                                     #用训练数据进行模型训练
runplt()
X2 =[[0],[20],[25],[30],[35],[50]]
#利用通过 fit()训练的模型对输入值的产出值进行预测
y2 =model.predict(X2)                              #预测数据
plt.plot(X,y,'k.')                                 #根据观察到的投入、产出值绘制点
plt.plot(X2,y2,'k-')                               #根据 X2、y2 绘制拟合的回归直线
plt.show()                                  #显示绘制的一元线性回归图,如图 5-1 所示
print('求得的一元线性回归方程的 b0 值为: %.2f'%model.intercept_)
print('求得的一元线性回归方程的 b1 值为: %.2f'%model.coef_)
print('预测投入 25 的产出值: %.2f'%model.predict([[25]]))     #输出投入 25 的预测值
```

运行上述代码,得到的输出结果如下:

```
求得的一元线性回归方程的 b0 值为: 18.95
求得的一元线性回归方程的 b1 值为: 1.18
预测投入 25 的产出值: 48.55
```

5.2.2 一元线性回归预测房价

波士顿房价数据集是由 D. Harrison 和 D. L. Rubinfeld 于 1978 年收集的波士顿郊区房屋的信息,数据集包含 506 行,每行包含 14 个字段,前 13 个字段(称为房屋的属性)用来描述房屋相关的各种信息,如周边犯罪率、是否在河边等相关信息,其中最后一个字段是房屋均价。14 个字段的具体描述如下。

CRIM:房屋所在镇的人均犯罪率。

ZN:用地面积超过 25 000 平方英尺(1 平方英尺=0.0929 平方米)的住宅所占比例。

INDUS:房屋所在镇无零售业务区域所占比例。

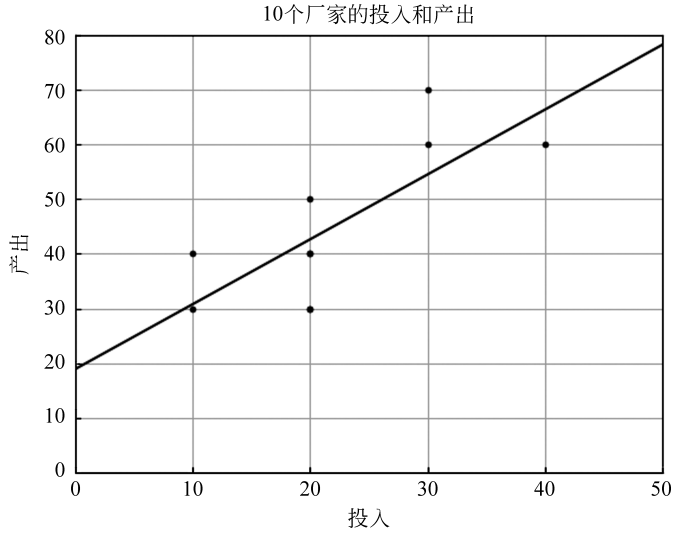

图 5-1　绘制的一元线性回归图

CHAS：是否邻近查尔斯河，1 是邻近，0 是不邻近。

NOX：一氧化氮浓度（千万分之一）。

RM：每处寓所的平均房间数。

AGE：自住且建于 1940 年前的房屋比例。

DIS：房屋距离波士顿五大就业中心的加权距离。

RAD：距离高速公路的便利指数。

TAX：每一万美元全额财产税金额。

PTRATIO：房屋所在镇的师生比。

B：城镇中黑人的比例。

LSTAT：人口中地位低下者所占的比例。

MEDV：自住房的平均房价（以 1000 美元为单位）。

【例 5-2】　使用 sklearn.linear_model 的 LinearRegression 模型拟合波士顿房价数据集。

在下面的内容中，我们将以房屋价格（MEDV）作为目标变量。

1）数据集的数据结构分析

```
>>>from sklearn.datasets import load_boston
>>>import pandas as pd
>>>import numpy as np
>>>import matplotlib.pyplot as plt                          #Python 中的绘图模块
>>>from sklearn.linear_model import LinearRegression       #导入线性回归模型
>>>boston=load_boston()                                    #加载波士顿房价数据集
>>>x=boston.data                                           #加载波士顿房价属性数据集
>>>y=boston.target                                         #加载波士顿房价数据集
>>>boston.keys()
```

```
dict_keys(['data', 'target', 'feature_names', 'DESCR'])
>>>x.shape
(506, 13)
>>>boston_df=pd.DataFrame(boston['data'],columns=boston.feature_names)
>>>boston_df['Target']=pd.DataFrame(boston['target'],columns=['Target'])
>>>boston_df.head(3)                                    #显示完整数据集的前 3 行数据
     CRIM     ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX   \
0 0.00632 18.0   2.31   0.0  0.538  6.575  65.2  4.0900  1.0  296.0
1 0.02731  0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242.0
2 0.02729  0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242.0

    PTRATIO       B  LSTAT  Target
0     15.3  396.90   4.98    24.0
1     17.8  396.90   9.14    21.6
2     17.8  392.83   4.03    34.7
```

2) 分析数据并可视化

房屋有 13 个属性,也就是说,13 个变量决定了房子的价格,这就需要计算这些变量和房屋价格的相关性。

```
>>>boston_df.corr().sort_values(by=['Target'],ascending=False)
             CRIM        ZN     INDUS      CHAS       NOX        RM       AGE  \
Target   -0.385832  0.360445 -0.483725  0.175260 -0.427321  0.695360 -0.376955
RM       -0.219940  0.311991 -0.391676  0.091251 -0.302188  1.000000 -0.240265
ZN       -0.199458  1.000000 -0.533828 -0.042697 -0.516604  0.311991 -0.569537
B        -0.377365  0.175520 -0.356977  0.048788 -0.380051  0.128069 -0.273534
DIS      -0.377904  0.664408 -0.708027 -0.099176 -0.769230  0.205246 -0.747881
CHAS     -0.055295 -0.042697  0.062938  1.000000  0.091203  0.091251  0.086518
AGE       0.350784 -0.569537  0.644779  0.086518  0.731470 -0.240265  1.000000
RAD       0.622029 -0.311948  0.595129 -0.007368  0.611441 -0.209847  0.456022
CRIM      1.000000 -0.199458  0.404471 -0.055295  0.417521 -0.219940  0.350784
NOX       0.417521 -0.516604  0.763651  0.091203  1.000000 -0.302188  0.731470
TAX       0.579564 -0.314563  0.720760 -0.035587  0.668023 -0.292048  0.506456
INDUS     0.404471 -0.533828  1.000000  0.062938  0.763651 -0.391676  0.644779
PTRATIO   0.288250 -0.391679  0.383248 -0.121515  0.188933 -0.355501  0.261515
LSTAT     0.452220 -0.412995  0.603800 -0.053929  0.590879 -0.613808  0.602339

              DIS       RAD       TAX   PTRATIO         B     LSTAT    Target
Target   0.249929 -0.381626 -0.468536 -0.507787  0.333461 -0.737663  1.000000
RM       0.205246 -0.209847 -0.292048 -0.355501  0.128069 -0.613808  0.695360
ZN       0.664408 -0.311948 -0.314563 -0.391679  0.175520 -0.412995  0.360445
B        0.291512 -0.444413 -0.441808 -0.177383  1.000000 -0.366087  0.333461
DIS      1.000000 -0.494588 -0.534432 -0.232471  0.291512 -0.496996  0.249929
CHAS    -0.099176 -0.007368 -0.035587 -0.121515  0.048788 -0.053929  0.175260
```

```
AGE       -0.747881  0.456022  0.506456  0.261515  -0.273534  0.602339  -0.376955
RAD       -0.494588  1.000000  0.910228  0.464741  -0.444413  0.488676  -0.381626
CRIM      -0.377904  0.622029  0.579564  0.288250  -0.377365  0.452220  -0.385832
NOX       -0.769230  0.611441  0.668023  0.188933  -0.380051  0.590879  -0.427321
TAX       -0.534432  0.910228  1.000000  0.460853  -0.441808  0.543993  -0.468536
INDUS     -0.708027  0.595129  0.720760  0.383248  -0.356977  0.603800  -0.483725
PTRATIO   -0.232471  0.464741  0.460853  1.000000  -0.177383  0.374044  -0.507787
LSTAT     -0.496996  0.488676  0.543993  0.374044  -0.366087  1.000000  -0.737663
```

从输出结果可以看出相关系数最高的是 RM，高达 0.69。也就是说，房间数和房价是强相关性。下面绘制房间数(RM)与房屋价格(MEDV)的散点图。

```
>>>import matplotlib
>>>matplotlib.rcParams['font.family'] = 'FangSong'      #设置中文字体格式为仿宋
>>>plt.scatter(boston_df['RM'],y)
<matplotlib.collections.PathCollection object at 0x000000001924C550>
>>>plt.xlabel('房间数(RM)',fontsize=15)
Text(0.5,0,'房间数(RM)')
>>>plt.ylabel('房屋价格(MEDV)',fontsize=15)
Text(0,0.5,'房屋价格(MEDV)')
>>>plt.title('房间数(RM)与房屋价格(MEDV)的关系',fontsize=15)
Text(0.5,1,'房间数(RM)与房屋价格(MEDV)的关系')
>>>plt.show()                          #显示绘制的房间数与房屋价格的散点图，如图 5-2 所示
```

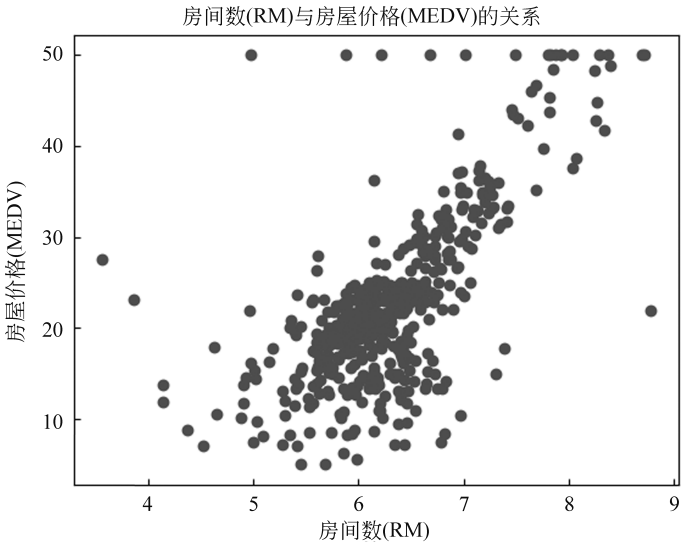

图 5-2　绘制的房间数与房屋价格的散点图

3）一元线性回归

(1) 去掉一些“脏”数据，比如去掉房价大于或等于 50 的数据和房价小于或等于 10 的数据。

```
>>>X=boston.data
>>>y=boston.target
>>>X=X[y<50]
>>>y=y[y<50]
>>>X=X[y>10]
>>>y=y[y>10]
>>>X.shape
(466, 13)
>>>y.shape
(466,)
```

(2) 构建线性回归模型。

```
>>>from sklearn.model_selection import train_test_split
#切分数据集,取数据集的 75%作为训练数据,25%作为测试数据
>>>X_train, X_test, y_train, y_test =train_test_split(X, y, random_state=1)
>>>LR =LinearRegression()
>>>LR.fit(X_train,y_train)
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

(3) 算法评估

```
>>>pre =LR.predict(X_test)
>>>print("预测结果", pre[3:8])                          #选取 5 个结果进行显示
预测结果 [27.48834701 21.58192891 20.36438243 22.980885 24.35103277]
>>>print(u"真实结果", y_test[3:8])                      #选取 5 个结果进行显示
真实结果 [22. 22. 24.3 22.2 21.9]
>>>LR.score(X_test,y_test)                             #模型评分
0.7155555361911698
```

这个模型的准确率只有 71.5%。

5.3 多元线性回归

5.3.1 多元线性回归模型

在一元线性回归中,将一个属性的样本集扩展到 d 个属性的样本集,即一个样本由 d 个属性描述,对 d 个属性的样本集的线性回归,拟合样本集得到一个模型:

$$f(\boldsymbol{x}_i)=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b$$

使得模型在 $\boldsymbol{x}_i$ 处的函数值 $f(\boldsymbol{x}_i)$接近 y_i,其中 $\boldsymbol{x}_i=[x_{i1},x_{i2},\cdots,x_{id}]^{\mathrm{T}}$ 为样本 d 个属性的列向量,$\boldsymbol{w}=[w_1,w_2,\cdots,w_d]^{\mathrm{T}}$ 为属性权重,这种多个属性的线性回归称为多元线性回归。

类似地,可利用最小二乘法对 $\boldsymbol{w}$ 和b 进行估计求值。为便于下面的讨论,我们将 b 合并到 $\boldsymbol{w}$,即 $\boldsymbol{w}=[w_1,w_2,\cdots,w_d]^{\mathrm{T}}$,相应地,每个样本 x_i 也增加一维,变为 $\boldsymbol{x}_i=[x_{i1},x_{i2},$

$\cdots, x_{id}]^{\mathrm{T}}$。

于是，用于求解多元线性回归参数的 $E(\boldsymbol{w},b)=\sum_{i=1}^{n}(y_i-(\boldsymbol{w}\boldsymbol{x}_i+b))^2$ 可以写成如下形式：

$$E(\boldsymbol{w})=(y-\boldsymbol{X}\boldsymbol{w})^{\mathrm{T}}(y-\boldsymbol{X}\boldsymbol{w})$$

其中 $\boldsymbol{y}$ 是样本的标记向量，$\boldsymbol{y}=[y_1,y_2,\cdots,y_n]^{\mathrm{T}}$，$\boldsymbol{X}$ 为样本矩阵。$\boldsymbol{X}$ 的形式如下。

$$\boldsymbol{X}=\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1d} & 1 \\ x_{21} & x_{22} & \cdots & x_{2d} & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nd} & 1 \end{bmatrix}=\begin{bmatrix} x_1^{\mathrm{T}} & 1 \\ x_2^{\mathrm{T}} & 1 \\ \vdots & \vdots \\ x_n^{\mathrm{T}} & 1 \end{bmatrix}$$

$E(\boldsymbol{w})=(y-\boldsymbol{X}\boldsymbol{w})^{\mathrm{T}}(y-\boldsymbol{X}\boldsymbol{w})$对参数 $\boldsymbol{w}$ 进行求导，求得的结果如下。

$$\frac{\partial E(\boldsymbol{w})}{\partial \boldsymbol{w}}=2\boldsymbol{X}^{\mathrm{T}}(X\boldsymbol{w}-y)$$

令 $2\boldsymbol{X}^{\mathrm{T}}(\boldsymbol{X}\boldsymbol{w}-y)$为零，可求得 $\boldsymbol{w}$ 的值为：

$$\boldsymbol{w}^{*}=(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

从 $\boldsymbol{w}^{*}=(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$ 可以发现 $\boldsymbol{w}^{*}$ 的计算涉及矩阵的求逆，只有在 $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ 为满秩矩阵或者正定矩阵时，才可以使用以上式子计算。但在现实任务中，$\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ 往往不是满秩矩阵，这样就会导致有多个解，并且这多个解都能使均方误差最小化，但并不是所有的解都适合做预测任务，因为某些解可能会产生过拟合的问题。

求出 $\boldsymbol{w}^{*}$ 后，线性回归模型的表达式为 $f(\boldsymbol{x}_i)=\boldsymbol{X}_i^{\mathrm{T}}(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$。

选择合适的自变量是正确进行多元回归预测的前提之一，多元回归模型自变量的选择可以利用变量之间的相关矩阵来解决。

多元线性回归模型可表示为：

$$y=\beta_0+\beta_1x_1+\beta_2x_2+\cdots+\beta_mx_m$$

回归模型中的回归参数(也称回归系数)$\beta_0,\beta_1,\cdots,\beta_m$ 利用样本数据估计，用得到的相应估计值 $b_0,b_1,\cdots,b_m$ 代替回归模型中的未知参数 $\beta_0,\beta_1,\cdots,\beta_m$，即得到估计的回归方程：

$$\hat{y}=b_0+b_1x_1+b_2x_2+\cdots+b_mx_m$$

下面给出使用 sklearn.linear_model 模块中的 LinearRegression 模型实现多元线性回归的例子。

【例 5-3】 求训练数据集 $X=[[0,0],[1,1],[2,2]]$、$y=[0,1,2]$的二元线性回归方程。

```
>>>from sklearn.linear_model import LinearRegression
>>>clf =LinearRegression()                                    #建立线性回归模型
>>>X =[[0,0],[1,1],[2,2]]
>>>y =[0,1,2]
>>>clf.fit(X,y)                                              #对建立的回归模型 clf 进行训练
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

```
>>>clf.coef_                                    #获取训练模型的 b1 和 b2
array([ 0.5, 0.5])
>>>clf.intercept_                               #获取训练模型的 b0
1.1102230246251565e-16
>>>clf.get_params()                             #获取训练所得的回归模型的参数
{'copy_X': True, 'fit_intercept': True, 'n_jobs': 1, 'normalize': False}
>>>clf.predict([[3, 3]])                        #使用训练得到的回归模型对[3,3]进行预测
array([ 3.])
>>>clf.score(X, y)                  #返回对以 X 为样本数据,以 y 为实际结果的预测效果评分
1.0
```

5.3.2 使用多元线性回归分析广告媒介与销售额之间的关系

为了增加商品销售量,商家通常会在电视、广播和报纸上进行商品宣传,在这 3 种媒介上的相同投入所带来的销售效果是不同的,如果能分析出广告媒体与销售额之间的关系,就可以更好地分配广告开支,并且使销售额最大化。

【例 5-4】 Advertising 数据集包含了 200 条不同市场的产品销售额,每个销售额对应 3 种广告媒体的投入,分别是 TV、radio 和 newspaper。Advertising 数据集前 8 条数据见表 5-2。

表 5-2 Advertising 数据集前 8 条数据

	TV	radio	newspaper	sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9
6	8.7	48.9	75	7.2
7	57.5	32.8	23. 5	11.8
8	120.2	19.6	11.6	13.2

下面给出多元线性回归分析广告媒介与销售额之间关系的代码。

```
import pandas as pd
import matplotlib
matplotlib.rcParams['font.family'] ='Kaiti'       #Kaiti 是中文楷体
from sklearn import linear_model
from sklearn.cross_validation import train_test_split  #这里引用了交叉验证
data =pd.read_csv('D:/Python/Advertising.csv')
feature_cols =['TV', 'radio', 'newspaper']         #指定特征属性
X =data[feature_cols]            #得到数据集的 3 个属性'TV' 'radio' 'newspaper'列
y =data['sales']                                   #得到数据集的目标列,即 sales 列
```